**Object Oriented Programming**

**Lab 7**

**Submitted To:**

Ma’am Amber Madeeha Zeb

**Submitted By:**

Manaal Waseem

FA18-BCE-074

**In Lab:**

**Task 1:**

**Imagine a publishing company that markets both book and audio-cassette versions of its works. Create a class publication that stores the *title* and *price* of a publication.**

**a. from this class derive two classes:**

**i. book, which adds a *page count* and**

**ii. tape, which adds a *playing time* in minutes.**

**iii. each of these three classes should have *getdata()* function to get its data from the user at the keyboard and a *putdata()* function to display its data.**

***b.* Write a main() program to test the book and tape class by creating instances of them, asking the user to fill in their data with *getdata()* and then displaying the data with *putdata().***

**Code:**

1 #include<iostream>

2 #include<conio.h>

3 #include<string>

4

5 **using namespace std**;

6

7 **class** Publication

8 {

9 **public**:

10 **void** get\_data();

11 **void** put\_data();

12 **private**:

13 **string** title;

14 **int** price;

15 };

16

17 **void** Publication::get\_data()

18 {

19 **cout** << "Enter the Title of Publication: " << **endl**;

20 **cin** >> title;

21 **cout** << "Enter the Price of Publication:" << **endl**;

22 **cin** >> price;

23 }

24

25 **void** Publication::put\_data()

26 {

27 **cout** << "Title of Publication: " << title <<**endl**;

28 **cout** << "Price of Publication: " << price << **endl**;

29 }

30

31 **class** Book:**public** Publication

32 {

33 **public**:

34 **void** get\_data();

35 **void** put\_data();

36 **private**:

37 **int** page\_count;

38 };

39

40 **void** Book::get\_data()

41 {

42 **cout** << "Enter the Number of Page: " << **endl**;

43 **cin** >> page\_count;

44 }

45

46 **void** Book::put\_data()

47 {

48 **cout** << "Number of Pages: " << page\_count << **endl**;

49 }

50

51 **class** Tape : **public** Publication

52 {

53 **public**:

54 **void** get\_data();

55 **void** put\_data();

56 **private**:

57 **int** playing\_time;

58 };

59

60 **void** Tape::get\_data()

61 {

62 **cout** << "Enter the playing time in minutes: " << **endl**;

63 **cin** >> playing\_time;

64 }

65

66 **void** Tape::put\_data()

67 {

68 **cout** << "Playing Time: " << playing\_time << **endl**;

69 }

70

71 **int** main ()

72 {

73 Book b1;

74 Tape t1;

75

76 b1.get\_data();

77 t1.get\_data();

78

79 b1.put\_data();

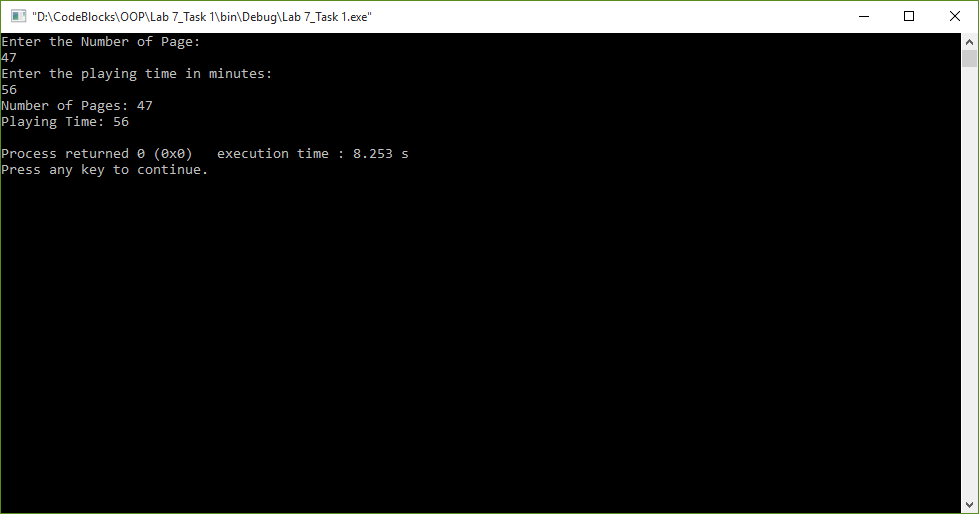
80 t1.put\_data();

81

82 **return** 0;

83 }

**Output:**



**Task 2:**

**Write a class Person that has attributes of *id, name* and *address*. It has a constructor to initialize, a member function to input and a member function to display data members. Create another class Student that inherits Person class. It has additional attributes of *rollnumber* and *marks*. It also has member function to input and display its data members.**

**Code:**

1 #include<iostream>

2 #include<conio.h>

3 #include<string>

4

5 **using namespace std**;

6

7 **class** Person

8 {

9 **public**:

10 Person()

11 {

12 id=0;

13 name='\0';

14 address='\0';

15 }

16 **void** get\_data();

17 **void** put\_data();

18 **private**:

19 **int** id;

20 **string** name, address;

21 };

22

23 **void** Person::get\_data()

24 {

25 **cout** << "Enter ID: " << **endl**;

26 **cin** >> id;

27 **cout** << "Enter Name: " << **endl**;

28 **cin** >> name;

29 **cout** << "Enter Address: " << **endl**;

30 **cin** >> address;

31 }

32

33 **void** Person:: put\_data()

34 {

35 **cout** << **endl** << "ID:" << id <<**endl**;

36 **cout** << "Name:" << name << **endl**;

37 **cout** << "Address:" << address << **endl**;

38 }

39

40 **class** Student : **public** Person

41 {

42 **public**:

43 **void** get\_data();

44 **void** put\_data();

45 **private**:

46 **int** roll\_number, marks;

47 };

48

49 **void** Student::get\_data()

50 {

51 Person::get\_data();

52

53 **cout** << **endl** << "Enter the Roll No: " << **endl**;

54 **cin** >> roll\_number;

55 **cout** << "Enter the Marks: " << **endl**;

56 **cin** >> marks;

57 }

58

59 **void** Student::put\_data()

60 {

61 Person::put\_data();

62

63 **cout** << "Roll No:" << roll\_number << **endl**;

64 **cout** << "Marks:" << marks << **endl**;

65 }

66

67 **int** main ()

68 {

69 Student s1;

70

71 s1.get\_data();

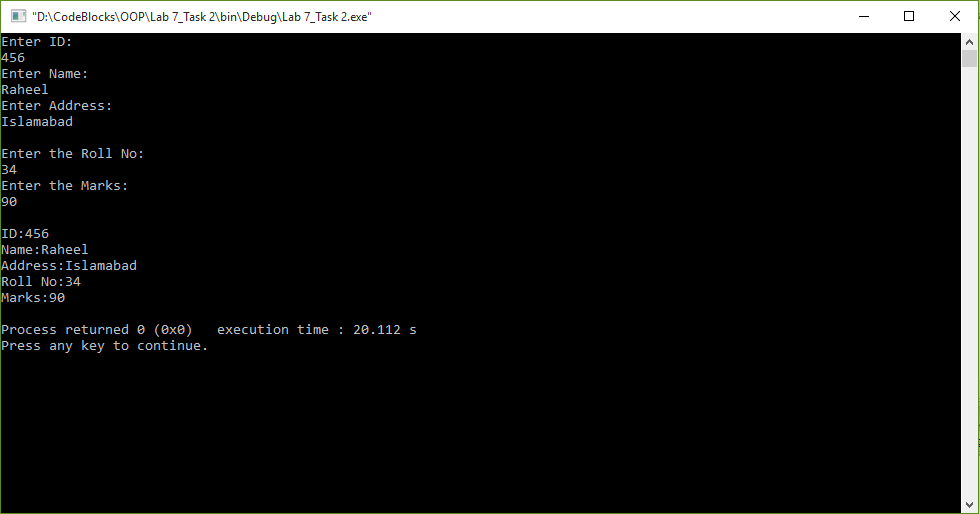
72 s1.put\_data();

73

74 **return** 0;

75 }

**Output:**



**Task 3:**

**Write a base class Computer that contains data members of *wordsize(in bits), memorysize (in megabytes), storagesize (in megabytes)* and *speed (in megahertz).* Derive a Laptop class that is a kind of computer but also specifies the object’s length, width, height, and weight. Member functions for both classes should include a default constructor, a constructor to inialize all components and a function to display data members.**

**Code:**

1 #include<iostream>

2 #include<conio.h>

3 #include<string>

4

5 **using namespace std**;

6

7 **class** Computer

8 {

9 **public**:

10 Computer():word\_size(0),memory\_size(0),storage\_size(0),speed(0){}

11 Computer(**int** w, **int** m, **int** s\_s, **float** s)

12 {

13 word\_size=w;

14 memory\_size=m;

15 storage\_size=s\_s;

16 speed=s;

17 }

18 **void** show\_data();

19

20 **private**:

21 **int** word\_size, memory\_size, storage\_size;

22 **float** speed;

23 };

24

25 **void** Computer::show\_data()

26 {

27 **cout**<< "Word Size:" << word\_size << "Bits" <<**endl**;

28 **cout**<< "Memory Size:" << memory\_size << "MB" <<**endl**;

29 **cout**<< "Storage Size:" << storage\_size << "MB" <<**endl**;

30 **cout**<< "Speed:" << speed << "MHz" <<**endl**;

31 }

32

33 **class** Laptop:**public** Computer

34 {

35 **public**:

36 Laptop():Computer(), length(0), width(0), height(0) ,weight(0){}

37 Laptop(**int** w, **int** m, **int** s\_s, **float** s, **float** l, **float** wid, **float** h, **float** weig):

38 Computer(w, m, s\_s, s)

39 {

40 length=l;

41 width=wid;

42 height=h;

43 weight=weig;

44 }

45 **void** show\_data();

46

47 **private**:

48 **float** length, width, height, weight;

49 };

50

51 **void** Laptop::show\_data()

52 {

53 Computer::show\_data();

54

55 **cout**<< "Length:" << length <<**endl**;

56 **cout**<< "Width:" << width <<**endl**;

57 **cout**<< "Height:" << height <<**endl**;

58 **cout**<< "Weight:" << weight <<**endl**;

59 }

60

61 **int** main ()

62 {

63 Laptop l1(4,4,4,4,4,4,44,4);

64

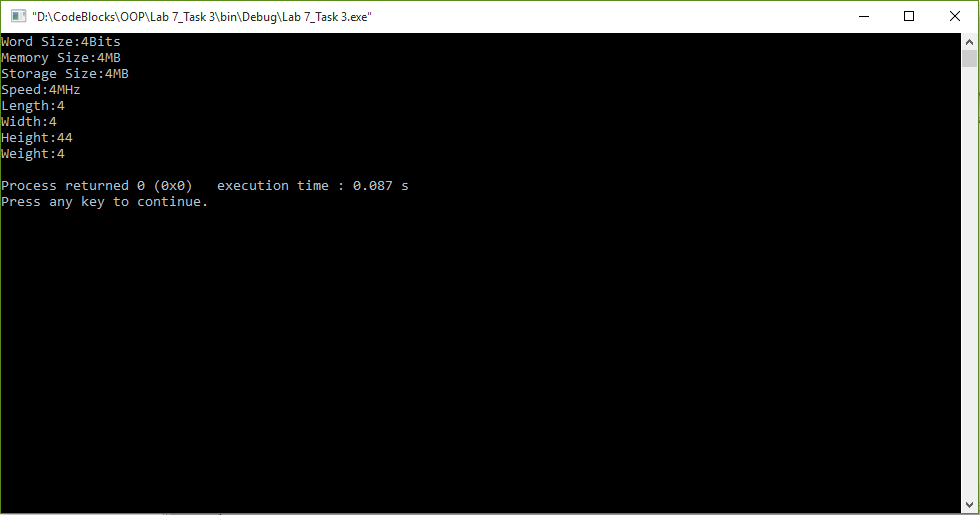
65 l1.show\_data();

66

67 **return** 0;

68 }

**Output:**



**Post Lab:**

**Task 1:**

**Write a program having a base class Student with data members *rollno, name* and Class define a member functions *getdata()* to input values and another function *putdata()* to display all values. A class Test is derived from class Student with data members *T1marks*, *T2marks*, *T3marks*, *Sessional1*, *Sessional2*, *Assignment* and *Final.* Also make a function *getmarks()* to enter marks for all variables except Final and also make a function *putmarks()* to display result. Make a function *Finalresult()* to calculate value for final variable using other marks. Then display the student result along with student data.**

**Code:**

1 #include<iostream>

2 #include<string>

3

4 **using namespace std**;

5

6 **class** Student

7 {

8 **public**:

9 **void** get\_data();

10 **void** put\_data();

11 **private**:

12 **int** roll\_no;

13 **string** name;

14 };

15

16 **void** Student::get\_data()

17 {

18 **cout** << **endl** << "Enter your Roll No: ";

19 **cin** >> roll\_no;

20

21 **cout** << **endl** << "Enter your name: ";

22 **cin** >> name;

23 }

24

25 **void** Student::put\_data()

26 {

27 **cout** << "Roll No:" << roll\_no <<**endl**;

28 **cout**<< "Name:" << name <<**endl**;

29 }

30

31 **class** Test : **public** Student

32 {

33 **public**:

34 **void** get\_marks();

35 **void** put\_marks();

36 **void** Finalresult();

37

38 **private**:

39 **float** T1marks, T2marks, T3marks, sessional1, sessional2, assignment ,**final**,

40 Tmarks,sum,percentage1,percentage2,result;

41 };

42

43 **void** Test::get\_marks()

44 {

45 Student::get\_data();

46

47 **cout** << **endl** << "Enter your marks:";

48 **cout**<< **endl** << "T1 marks:";

49 **cin** >> T1marks;

50 **cout**<< **endl** << "T2 marks:";

51 **cin** >> T2marks;

52 **cout**<< **endl** << "T3 marks:";

53 **cin** >> T3marks;

54 **cout**<< **endl** << "Sessional 1:";

55 **cin** >> sessional1;

56 **cout**<< **endl** << "Sessional 2:";

57 **cin** >> sessional2;

58 **cout**<< **endl** << "Assignment:";

59 **cin** >> assignment;

60 **cout**<< **endl** << "Final:";

61 **cin** >> **final**;

62 }

63

64 **void** Test::put\_marks()

65 {

66 Student::put\_data();

67

68 **cout** << "T1 marks:"<< T1marks <<**endl**;

69 **cout** << "T2 marks:"<< T2marks <<**endl**;

70 **cout** << "T3 marks:"<< T3marks <<**endl**;

71 **cout** << "Sessional 1:" << sessional1 <<**endl**;

72 **cout** << "Sessional 2:" << sessional2 <<**endl**;

73 **cout** << "Assignment:" << assignment <<**endl**;

74 **cout** << "Final:" << **final** <<**endl**;

75 }

76

77 **void** Test::Finalresult()

78 {

79 Tmarks=((T1marks+T2marks+T3marks)/30)\*15;

80 sum=Tmarks+assignment+sessional1+sessional2;

81 percentage1=(sum/50)\*100;

82 percentage2=(**final**/50)\*100;

83 result=(percentage1+percentage2)/2;

84 **cout** << "Final result:" << result<< "%";

85 }

86

87 **int** main()

88 {

89 Test t1;

90 t1.get\_marks();

91

92 **cout**<<**endl**<<**endl**;

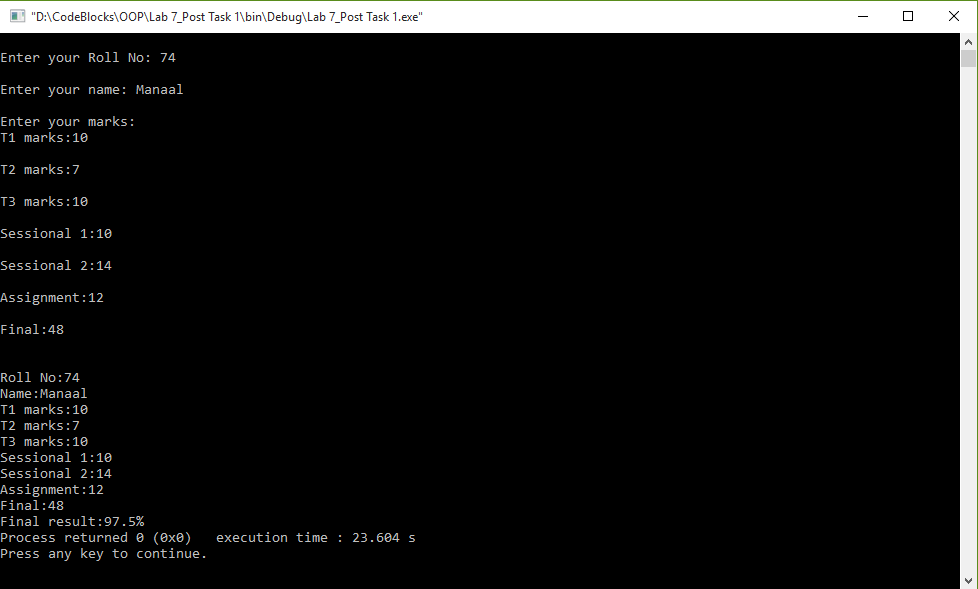
93

94 t1.put\_marks();

95 t1.Finalresult();

96 }

**Output:**



**Task 2:**

**Write a program that declares two classes. The parent class is called *Simple* that has two data members *num1* and *num2* to store two numbers. It also has four member functions.**

* **The *add()* function adds two numbers and displays the result.**
* **The *sub()* function subtracts two numbers and displays the result.**
* **The *mul()* function multiplies two numbers and displays the result.**
* **The *div()* function divides two numbers and displays the result.**

**The child class is called *Complex* that overrides all four functions. Each function in the child class checks the value of data members. It calls the corresponding member function in the parent class if the values are greater than 0. Otherwise it displays error message.**

**Code:**

1 #include<iostream>

2 #include<string>

3

4 **using namespace std**;

5

6 **class** Simple

7 {

8 **protected**:

9 **float** num1, num2, result;

10

11 **public**:

12 **void** get\_data();

13 **void** add();

14 **void** sub();

15 **void** mul();

16 **void** div();

17 };

18

19 **void** Simple::get\_data()

20 {

21 **cout** << **endl** << "Enter First Number:";

22 **cin** >> num1;

23

24 **cout** << **endl** << "Enter Second Number:";

25 **cin** >> num2;

26 }

27

28 **void** Simple::add()

29 {

30 result=num1+num2;

31 **cout**<< "Result: " << result <<**endl**;

32 }

33

34 **void** Simple::sub()

35 {

36 result=num1-num2;

37 **cout**<< "Result: " << result <<**endl**;

38 }

39

40 **void** Simple::mul()

41 {

42 result=num1\*num2;

43 **cout**<< "Result: " << result <<**endl**;

44 }

45

46 **void** Simple::div()

47 {

48 result=num1/num2;

49 **cout**<< "Result: " << result <<**endl**;

50 }

51

52 **class** Complex:**public** Simple

53 {

54 **public**:

55 **void** add();

56 **void** sub();

57 **void** mul();

58 **void** div();

59 };

60

61 **void** Complex::add()

62 {

63 **if**(num1>0 && num2>0)

64 Simple::add();

65 **else**

66 **cout**<< "ERROR, Please try again.." <<**endl**;

67 }

68

69 **void** Complex::sub()

70 {

71 **if**(num1>0 && num2>0)

72 Simple::sub();

73 **else**

74 **cout**<< "ERROR, Please try again.." <<**endl**;

75 }

76

77 **void** Complex::mul()

78 {

79 **if**(num1>0 && num2>0)

80 Simple::mul();

81 **else**

82 **cout**<< "ERROR, Please try again.." <<**endl**;

83 }

84

85 **void** Complex::div()

86 {

87 **if**(num1>0 && num2>0)

88 Simple::div();

89 **else**

90 **cout**<< "ERROR, Please try again.." <<**endl**;

91 }

92

93 **int** main()

94 {

95 Complex c1;

96

97 c1.get\_data();

98

99 c1.add();

100 c1.sub();

101 c1.mul();

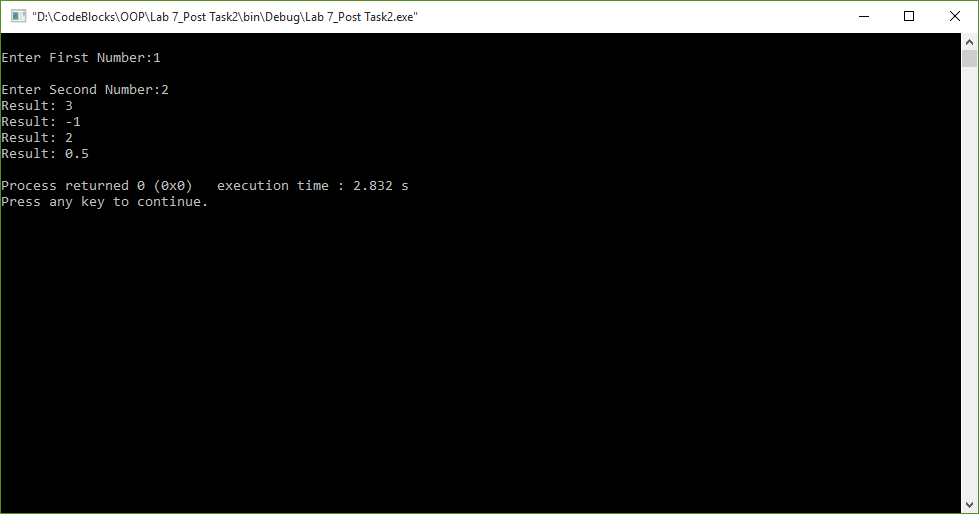
102 c1.div();

103

104 **return** 0;

105 }

**Output:**



**Task 3:**

**An electricity board charges the following rates to domestic users to discourage large consumption of energy.**

* **For the first 100 units − 50 P per unit**
* **Beyond 100 units − 60 P per unit**

**If the total *cost* is more than Rs.250.00 then an additional surcharge of 15% is added on the difference. Define a class Electricity in which the function Bill computes the *cost*. Define a derived class More\_Electricity and override Bill to add the *surcharge*.**

**Code:**

1 #include <iostream>

2 #include <string>

3

4 **using namespace std**;

5

6 **class** Electricity

7 {

8 **public**:

9 **void** get\_user\_info();

10 **void** disp\_user\_info();

11 **float** Bill();

12

13 **private**:

14 **string** user\_name;

15 **int** unit;

16 };

17

18 **void** Electricity::get\_user\_info()

19 {

20 **cout**<< "Enter the Name of User: " <<**endl**;

21 **cin**>> user\_name;

22 **cout**<< "Enter Units used by User: " <<**endl**;

23 **cin**>> unit;

24 }

25

26 **void** Electricity::disp\_user\_info()

27 {

28 **cout**<< "Name" << "\t" << user\_name <<**endl**;

29 **cout**<< "Units Utilized" << "\t" << unit <<**endl**;

30 }

31

32 **float** Electricity::Bill()

33 {

34 **if**(unit<=100)

35 {

36 **return** (0.5\*unit);

37 }

38 **else**

39 **if**(unit>100)

40 {

41 **return** (0.6\*unit);

42 }

43 }

44

45 **class** More\_Electricity:**public** Electricity

46 {

47 **public**:

48 **void** Bill();

49 **void** Bill\_disp();

50

51 **private**:

52 **float** difference;

53 **float** cost;

54 };

55

56 **void** More\_Electricity::Bill()

57 {

58 Electricity::get\_user\_info();

59 cost=Electricity::Bill();

60

61 **if**(cost>250)

62 {

63 difference=cost-250;

64 cost=(0.15\*difference)+cost;

65 }

66 }

67

68 **void** More\_Electricity::Bill\_disp()

69 {

70 Electricity::disp\_user\_info();

71

72 **cout**<< "Bill" << "\t" << cost <<**endl**;

73 }

74

75 **int** main()

76 {

77 More\_Electricity m1;

78

79 m1.Bill();

80 m1.Bill\_disp();

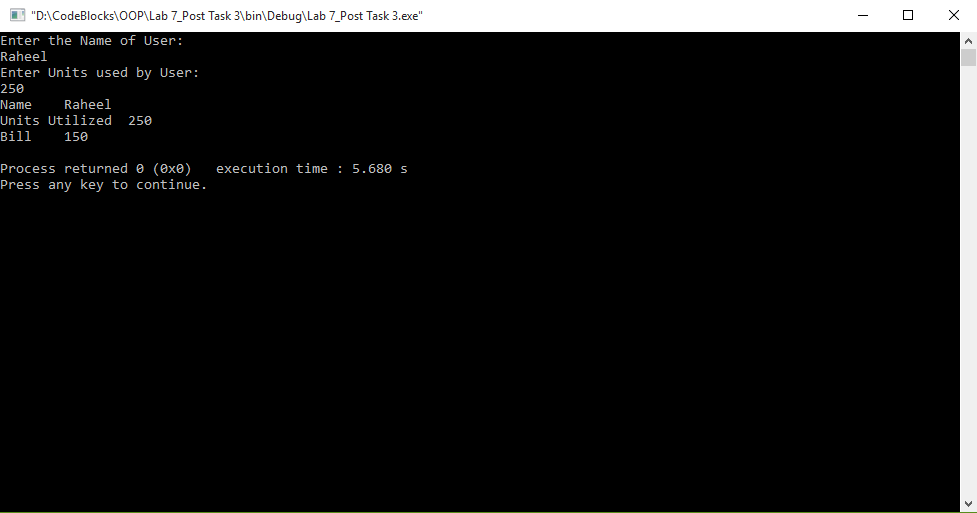
81

82 **return** 0;

83 }

84

**Output:**



**Task 4:**

**(Package Inheritance Hierarchy) Package-delivery services, such as FedEx®, DHL® and UPS®, offer a number of different shipping options, each with specific costs associated. Create an inheritance hierarchy to represent various types of packages. Use Package as the base class of the hierarchy, then include classes TwoDayPackage and OvernightPackage that derive from Package. Base class Package should include data members representing the *name, address, city, state* and *ZIP code* for both the sender and the recipient of the package, in addition to data members that store the *weight* (in ounces) and *cost per ounce* to ship the package. Package's constructor should initialize these data members. Ensure that the *weight* and *cost per ounce* contain positive values. Package should provide a public member function *calculateCost*() that returns a double indicating the cost associated with shipping the package. Package's *calculateCost()* function should determine the cost by multiplying the *weight* by the *cost per ounce*. Derived class TwoDayPackage should inherit the functionality of base class Package, but also include a data member that represents a *flat fee* that the shipping company charges for two-day-delivery service. TwoDayPackage's constructor should receive a value to initialize this data member. TwoDayPackage should redefine member function *calculateCost*() so that it computes the *shipping cost* by adding the *flat fe*e to the *weight-based cost* calculated by base class Package's *calculateCost*() function. Class OvernightPackage should inherit directly from class Package and contain an additional data member representing an additional fee per ounce charged for overnight-delivery service. OvernightPackage should redefine member function calculateCost() so that it adds the additional *fee per ounce* to the standard *cost per ounce* before calculating the shipping *cost*. Write a test program that creates objects of each type of Package and tests member function *calculateCost*().**

**Code:**

1 #include <iostream>

2 #include <string>

3

4 **using namespace std**;

5

6 **class** Package

7 {

8 **public**:

9 Package(**string** s\_n, **string** r\_n, **string** s\_a, **string** r\_a, **string** s\_c, **string** r\_c, **string** s\_s, **string** r\_s, **string** s\_Z, **string** r\_Z, **double** w, **double** c)

10 {

11 s\_name = s\_n;

12 r\_name = r\_n;

13 s\_address = s\_a;

14 r\_address = r\_a;

15 s\_city = s\_c;

16 r\_city = r\_c;

17 s\_state = s\_s;

18 r\_state = r\_s;

19 s\_ZIPcode = s\_Z;

20 r\_ZIPcode = r\_Z;

21 **if** (w>0 && c>0)

22 {

23 weight = w;

24 cost\_oz = c;

25 }

26 **else**

27 **cout**<< "ERROR" <<**endl**;

28 }

29

30 **double** calculateCost();

31 **double** cost\_ounce();

32 **double** weight\_oz();

33 **void** disp();

34

35 **private**:

36 **string** s\_name, r\_name, s\_address, r\_address, s\_city, r\_city, s\_state,

37 r\_state, s\_ZIPcode, r\_ZIPcode;

38 **double** weight, cost\_oz;

39 };

40

41 **double** Package::calculateCost()

42 {

43 **return** (cost\_oz\*weight);

44 }

45

46 **double** Package::cost\_ounce()

47 {

48 **return** cost\_oz;

49 }

50

51 **double** Package::weight\_oz()

52 {

53 **return** weight;

54 }

55

56 **void** Package::disp()

57 {

58 **double** temp=calculateCost();

59

60 **cout**<< "Cost of Package= " << temp <<**endl**;

61 }

62

63 **class** TwoDayPackage:**public** Package

64 {

65 **public**:

66 TwoDayPackage(**string** s\_n, **string** r\_n, **string** s\_a, **string** r\_a, **string** s\_c, **string** r\_c, **string** s\_s,

67 **string** r\_s, **string** s\_Z, **string** r\_Z, **double** w, **double** c, **double** f) : Package(s\_n, r\_n, s\_a, r\_a, s\_c, r\_c, s\_s, r\_s, s\_Z, r\_Z, w, c)

68 {

69 flat\_fee=f;

70 }

71

72 **double** calculateCost();

73 **void** disp();

74

75 **private**:

76 **double** flat\_fee;

77 };

78

79 **double** TwoDayPackage::calculateCost()

80 {

81 **double** temp=Package::calculateCost();

82 **return** (temp+flat\_fee);

83 }

84

85 **void** TwoDayPackage::disp()

86 {

87 **double** temp=calculateCost();

88

89 **cout**<< "Cost of Two Day Package= " << temp <<**endl**;

90 }

91

92 **class** OvernightPackage:**public** Package

93 {

94 **public**:

95 OvernightPackage(**string** s\_n, **string** r\_n, **string** s\_a, **string** r\_a, **string** s\_c, **string** r\_c, **string** s\_s,

96 **string** r\_s, **string** s\_Z, **string** r\_Z, **double** w, **double** c, **double** o) : Package(s\_n, r\_n, s\_a, r\_a, s\_c, r\_c, s\_s, r\_s, s\_Z, r\_Z, w, c)

97 {

98 feePerOunce=o;

99 }

100

101 **double** calculateCost();

102 **void** disp();

103

104 **private**:

105 **double** feePerOunce;

106 };

107

108 **double** OvernightPackage::calculateCost()

109 {

110 **double** temp1=Package::cost\_ounce();

111 **double** temp2=Package::weight\_oz();

112

113 feePerOunce+=temp1;

114 **return** (feePerOunce\*temp2);

115 }

116

117 **void** OvernightPackage::disp()

118 {

119 **double** temp=calculateCost();

120

121 **cout**<< "Cost of Overnight Package= " << temp <<**endl**;

122 }

123

124 **int** main()

125 {

126 Package p1("Tahir", "Yasir", "H10A", "H45B", "Islamabad", "Karachi", "Pakistan", "Pakistan", "4404", "5607", 350, 450);

127 TwoDayPackage td1("Tahir", "Yasir", "H10A", "H45B", "Islamabad" ,"Karachi", "Pakistan", "Pakistan", "4404", "5607", 350, 450, 256);

128 OvernightPackage ov1("Tahir", "Yasir", "H10A", "H45B", "Islamabad", "Karachi", "Pakistan", "Pakistan", "4404", "5607", 350, 450, 678);

129

130 p1.calculateCost();

131 p1.disp();

132 td1.calculateCost();

133 td1.disp();

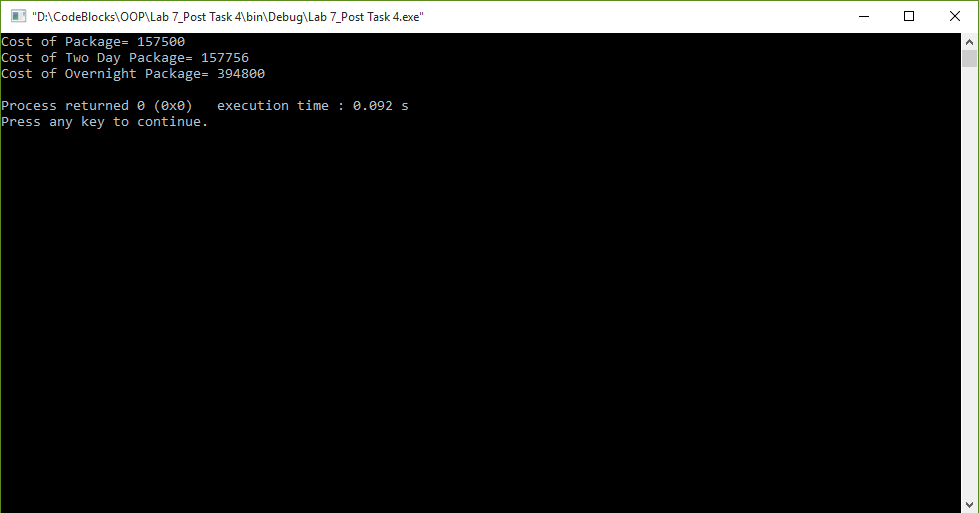
134 ov1.disp();

135

136 **return** 0;

137 }

**Output:**



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**THE END**